Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2011, Article ID 951310, 13 pages
doi:10.1155/2011/951310

Research Article

A DiffServ Architecture for QoS-Aware Routing for Delay-Sensitive and Best-Effort Services in IEEE 802.16 Mesh Networks

Ishita Bhakta,[1] Sandip Chakraborty,[2] Barsha Mitra,[3] Debarshi Kumar Sanyal,[3] Samiran Chattopadhyay,[3] and Matangini Chattopadhyay[4]

[1] *School of Mobile Computing and Communications, Jadavpur University, Kolkata 700032, India*
[2] *Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, India*
[3] *Department of Information Technology, Jadavpur University, Kolkata 700032, India*
[4] *School of Education Technology, Jadavpur University, Kolkata 700032, India*

Correspondence should be addressed to Debarshi Kumar Sanyal, debarshisanyal@gmail.com

Received 11 April 2011; Accepted 11 August 2011

Academic Editor: Youyun Xu

In today's networks, the widespread use of real-time traffic such as video and audio applications demand special service guarantee in terms of throughput, delay, and jitter, thus making quality of service (QoS) a key problem. IEEE 802.16 mesh networks are likely to be the basis of next-generation last mile network connectivity. So, providing QoS is one of the major designing goals in IEEE 802.16 mesh network. While the standard defines five service classes for PMP mode, no standard defined service classes exist for mesh mode. In this paper, we describe a differentiated service (DiffServ) architecture for QoS support in IEEE 802.16 mesh network by considering a basic requirement for QoS guarantee—delay. A new cross-layer routing metric is proposed, namely, expected scheduler delay (ESD). An efficient distributed scheme is proposed to calculate ESD and route the packets using source routing mechanism. This scheme is capable of differentiating between delay sensitive and best-effort traffic and route packets accordingly. Finally, the results of the proposed scheme are compared with the standard schemes that take hop count as a routing metric.

1. Introduction

IEEE 802.16 standard, known as the "Air Interface for Fixed Broadband Wireless Access System" [1] is the next generation *wireless metropolitan area network* (WMAN) that aims at providing last mile wireless broadband connectivity. The IEEE 802.16 technology can operate in two modes—*point to multipoint* (PMP) and *mesh mode*. The mesh mode architecture of IEEE 802.16 network can operate in both centralized and distributed manner. The distributed mesh architecture uses relay nodes for better spatial reuse and larger connectivity through multihop communication.

In today's heterogeneous network, several types of traffic such as voice, video and data demand different types of performance assurance and service differentiation, and so QoS provisioning is one of the important goals in the design of next-generation networks. The PMP mode of IEEE 802.16 supports five types of services—*unsolicited grant services* (UGS) to support CBR or CBR-like flows such as VoIP, *real-time polling service* (rtPS) for real time VBR-like flows such as MPEG video, *non real time polling service* (nrtPS) such as bandwidth-intensive file transfer, *extended real time polling service* (ertPS) for better support of VoIP, and finally the *best effort* (BE) service for ftp file transfer, web browsing and so forth. These service classes are based on bandwidth requirements by the *subscriber stations* (SSs) for uplink and downlink scheduling, and the *base station* (BS) manages the bandwidth allocation-based on the specific service classes in which the flow belongs. There are several problems with such bandwidth allocation based service classes in case of mesh network. First, there is no BS in distributed mesh, and so the bandwidth allocation algorithm should be fully distributed among the nodes. Second, as there are multiple paths from source to destination, routing is a big issue. The traditional

802.16 mesh network uses standard Dijkstras shortest path routing algorithm which is not very well suited in providing QoS.

Delay is one of the main requirements for quality delivery of real time traffic such as voice and video. So, such traffic should be routed along a minimum delay path. In today's network, there are mainly two types of QoS architectures—Differentiated Service Architecture (DiffServ) and Integrated Service Architecture (IntServ). The PMP mode uses IntServ architecture by providing five service classes. In this case the QoS is provided on per-flow basis, by reserving the required resources to satisfy QoS requirements. However, as described earlier, this architecture is not well suited in mesh mode due to the lack of a centralized resource allocator. So, the DiffServ architecture can be used here by designing a routing metric that takes several network delays (such as scheduling delay congestion delay) in consideration.

Several routing metrics have been proposed in the literature for quality delivery of traffic—such as Expected Transmission Count (ETX), Expected Transmission Time (ETT), and Minimum Loss (ML). However, all these routing metrics are implicitly catered to IEEE 802.11-based mesh networks. IEEE 802.16 mesh has inherent difference with IEEE 802.11 mesh. These routing metrics calculate the required values using two-way MAC handshaking (DATA-ACK) procedure whereas 802.16 distributed mesh does not use this handshaking procedure, and the MAC protocol is fundamentally different. So, these routing metrics cannot be used directly here. Furthermore the large network architecture and scalability issues in IEEE 802.16 mesh demand a new cross layer routing metric that takes scheduling procedure into account.

In [2], the authors have proposed a mathematical model for the performance of IEEE 802.16 mesh distributed coordinated scheduler. From the mathematical model proposed by them, it can be seen that the scheduling parameter, called *hold-off exponent*, that determines the minimum number of transmission slots a node must wait between two successful transmissions, has a great impact on scheduler performance. A suitably chosen hold-off exponent can improve the scheduler performance significantly by minimizing the number of waiting slots between two transmissions, and hence reduces the scheduling delay. This mathematical expectation is used to find out the delay of several paths from a source to a destination, and the most suitable path for delay sensitive traffic is chosen, which is the one having the minimum path delay between a source and a destination.

In this paper an architecture is described where the most optimized path is selected for traffic requirements based on minimum delay prediction along the path. Our contributions of this paper are in four aspects.

(i) A cross layer routing metric is presented, namely, *expected scheduler delay (ESD)*, based on MAC scheduler for quality delivery of delay sensitive traffics.

(ii) A modification of Bellman-Ford shortest path routing algorithm is proposed to choose the next best hop for routing a packet across the network. We also propose a scheme for coordination between the nodes; such that they have sufficient knowledge to calculate the cross layer routing metric, ESD.

(iii) A differentiated service architecture is introduced where the packets are routed across the network depending on whether the traffic is delay-sensitive or not. Delay-sensitive traffic employs ESD as the routing metric whereas the best effort one considers the traditional hop count metric for routing.

(iv) Finally we perform extensive simulation using NS2 network simulator, to show that ESD can find out a minimum delay path, and performs dramatically better than hop count in case of high load. We also compare the performance of delay sensitive service based on the ESD metric with the best effort one based on the hop count metric to illustrate that the delay-sensitive service meets the delay requirements for realtime traffic.

2. Related Works

Several works exist in the literature for QoS provisioning in IEEE 802.16 PMP mode, but very few of them focus on the mesh mode. This is because the mesh mode is relatively new, and several of its features are under development and research. In [3], the authors describe the standard defined service classes for IEEE 802.16 in PMP mode and their implementations. They have shown the relationship between traffic characteristics and its QoS requirements and network performance for the four service classes BE, rtPS, nrtPS, and UGS. The ertPS service class is a recent addition to IEEE 802.16 PMP mode to support better VBR realtime traffic such as VoIP. In [4, 5], the QoS provisioning mechanisms of IEEE 802.16 mesh network have been studied. In [4], the authors describe a *fair end-to-end bandwidth allocation algorithm (FEBA)* based on centralized scheduling in 802.16 mesh network, where the mesh-BS allocates bandwidth based on the traffic requirements. However, none of these algorithms support QoS in mesh distributed scheduler. Supporting QoS in distributed mesh is more challenging than centralized one because of the lack of any centralized coordinator. All the above mentioned works take into account scheduler performance to minimize delay, but none of these consider the routing of packets for delay sensitive services.

Li et al. [6] focus on the end-to-end delay factor in case of routing for delay sensitive services. Their approach selects the next hop dynamically in a greedy manner and computes routing paths having minimum end-to-end delays. This routing algorithm is implemented at the medium access control (MAC) layer. Each node finds out the next hop for the passing flows using the scheduling information and tries to forward packets in the earliest slots. Also, a loop cancellation mechanism to ensure loop-free routing is proposed. The authors have shown that this approach is capable of lowering the delay of traffic flows and also achieving load balancing to a certain extent. In [7], Tsai and Wang introduced a routing and admission control mechanism in IEEE 802.16 mesh distributed network based on the bandwidth estimation and

token bucket admission control. The token bucket is used to control the traffic patterns for the purpose of estimating the bandwidth requirements of a connection. The proposed TAC algorithm performs bandwidth estimation by considering the hop count and delay requirements of realtime traffic flows. It guarantees the delay requirements of realtime traffic, and avoids starvation of low-priority traffic to establish a QoS-enabled environment.

As mentioned earlier, differentiated service architecture and integrated service architecture are the two main QoS architectures prevalent in today's networks. Several contemporary works focus on these models. In [8], the authors discuss the general architecture of integrated service (IntServ) and differentiated service (DiffServ) for the Internet. While the IntServ architecture reserves resources for each flow, DiffServ architecture guarantees QoS by classifying packets at ingress and egress routers inside a differentiated service domain. The wireless DiffServ defines two service classes, besides the traditional best-effort service—premium and assured. In order to forward traffic in the core networks, the expedited forwarding (EF) per-hop-behavior is applied for the premium service, and the assured forwarding (AF) per-hop behavior is applied for assured service. In [9], Jiang et al. propose a differentiated service architecture (DiffServ) for wireless mesh backbone. The proposed DiffServ has higher scalability and small signaling overhead. In [10], the authors divide the network into nonoverlapping clusters, each cluster representing an interference domain. They suggest a two-level routing scheme intercluster routing and intracluster routing. Intercluster routing performs routing of data cluster by cluster whereas the intra-cluster routing performs the routing of data inside the cluster. It is efficient in terms of scalability but lacks in terms of interference and thus throughput and delay.

A literature survey of the existing works on wireless mesh networks such as [11, 12] reveal that a number of routing metrics are available for routing in wireless mesh networks based on IEEE 802.11. Among those metrics hop count (HC), expected transmission count (ETX), expected transmission time (ETT), and minimum loss (ML) are a few. Hop count is the most common routing metric used in both wired and wireless networks. It reflects the effect of path lengths on the performance of flows. However, it does not consider the differences in transmission rates, packet losses in wireless links, interference among traffic flows, presence of multiple channels, and variations of load among different paths. ETX is one of the first metrics specifically proposed for IEEE 802.11 ad hoc networks. The ETX of a wireless link represents the expected number of data transmissions required to send a packet over that link to another node, including retransmissions. The ETX of a route is the sum of the ETX of each link along the route. However, it does not consider link interference and the differences in link transmission rates. The ETT metric was proposed as an improvement over ETX. ETT takes in account the differences in transmission rates of the links, thus overcoming the shortcoming of ETX. The ETT of a link l is defined as the expected time required to successfully transmit a packet over a link l. The weight of a path is the summation of the ETTs of the links along that path. The minimum loss (ML) metric computes the most suitable path by optimizing packet delivery ratio. ML finds the route where the probability of end-to-end packet loss is very low.

Table 1: Comparison of Routing Metrics.

Characteristics	HC	ETX	ETT	ML	ESD
Quality-Aware	×	✓	✓	✓	✓
Cross-Layer Support	×	×	×	×	✓
Link-load consideration	×	×	×	×	✓
Delay consideration	×	×	×	×	✓
Congestion Avoidance	×	×	×	×	✓

The above-mentioned metrics, namely, HC, ETX, ETT, and ML were designed for routing primarily in IEEE 802.11 networks. They calculate the required parameters mainly using the ACK frames used for DCF handshaking. The DATA-ACK-based handshaking is not used in IEEE 802.16 mesh cooperative scheduling. Also, IEEE 802.16 mesh scheduler introduces extra scheduling delay. In addition to these, the IEEE 802.16 mesh networks being much larger in size than their IEEE 802.11 counterparts, congestion is a crucial factor to be considered in 802.16 mesh networks. So, the above-mentioned metrics are not directly applicable to IEEE 802.16 mesh network. In this paper, a new cross layer based routing metric is proposed, namely *expected scheduler delay (ESD)* that takes scheduler delay and congestion into account. A comparison of ESD with the other metrics is shown in Table 1.

In [2], the authors formulate a mathematical model for 802.16 mesh distributed coordinated scheduler, and analyze the effect of hold-off exponent on the performance of mesh distributed coordinated scheduler. They have calculated a fixed point iterative equation for mathematical expectation of number of slots between two successful control message (MSH-DSCH message) transmissions, and shown that the data transmission and the overall performance of the system depends completely on the scheduling of control messages. Based on this model, Bayer et al. [13] propose a node differentiation scheme using the effect of hold-off exponents on the scheduler performance. They have differentiated the nodes in a mesh into four classes: mesh bs, active node (AN) which transmit and receive data, sponsoring node (SN), the parents of ANs, and Inactive Node (IAN), which are not part of any one of the above classes. Based on the bandwidth requirement of the nodes, the authors propose a scheme to dynamically adjust the hold-off exponents of the nodes such that

$$\mathrm{EXP}_{\mathrm{BS}} \leq \mathrm{EXP}_{\mathrm{AN}} \leq \mathrm{EXP}_{\mathrm{SN}} \leq \mathrm{EXP}_{\mathrm{IAN}}. \quad (1)$$

And so the bandwidth requirement of mesh BS is highest while the bandwidth requirement for inactive nodes is the least. However, this scheme requires excessive analysis of each node's traffic, and the convergence time is high. In [14], Wang et al. derive a dynamic hold-off exponent adjustment algorithm based on the number of two-hop neighbors and the data traffic analysis. They have proposed a two-phase scheme for setting hold-off time in order to improve

the network utilization. They delineate two approaches to acheiving this adjustment of hold-off exponents: static and dynamic. According to their results both approaches perform better in terms of increasing the utilization of the control-plane bandwidth and decreasing the time required to establish data schedules than the standard algorithm. Also, both the approaches provide efficient and fair scheduling for IEEE 802.16 mesh networks, thereby giving good application performance. In [15], the authors propose a scheme for tuning hold-off exponent, an important parameter for mesh distributed scheduling for minimizing the delay at the network. They propose a distributed search protocol which is used by each node to select the hold-off exponent that minimizes the expected delay between its two successive control channel transmissions. They have shown that this scheme improves the network performance in terms of end-to-end delay and throughput.

In a previous work [16], we have given a primary introduction to the proposed ESD metric. However, there has been no formal performance and quality analysis of the proposed metric, and only a limited number of simulation results have been given to show its performance over the standard routing procedure. This paper describes the metric more formally, and analyzes it using the basic requirements of a routing metric for wireless mesh networks. Furthermore this paper also introduces a service differentiation scheme between delay sensitive and best effort services, using the proposed routing metric. Extensive simulations show that the system substantially improves QoS requirement for delay sensitive services while keeping the performance of best effort services at per the standard.

3. IEEE 802.16 Mesh Scheduling

The IEEE 802.16 mesh mode scheduling is of two types: *centralized* and *distributed*. In centralized scheduling, there is a base station (mesh coordinator) that establishes network-wide schedules and manages bandwidth allocation among the subscriber stations (SSs). In contrast, the distributed scheduling protocol negotiates pairwise bandwidth assignments between mesh routers. In this mode of scheduling there is no centralized coordinator and each SS (node) runs the scheduling algorithm in a distributed manner. Distributed scheduling can again be classified into two groups—*coordinated* and *uncoordinated*. In the coordinated distributed scheduling mode, the scheduling messages are passed among the SSs, and in the uncoordinated distributed scheduling mode, there is no such coordination. Here we focus only on the distributed coordinated scheduling. The scheduling messages mentioned above are special types of messages, mainly MSH-DSCH and MSH-NCFG messages. The MSH-DSCH message is used to transmit distributed scheduling information and the MSH-NCFG message contains network setup and mesh management information. Here our primary focus is on MSH-DSCH. Whenever a node transmits the MSH-DSCH message, it includes certain parameters of the neighbors in the message so that every node has the knowledge of the schedule information of its two-hop neighborhood. One such parameter is called transmit hold-off exponent (TxtHoldoffExp). This parameter determines the number of transmission opportunities a node must wait before next contention after a successful transmission. The number of such transmission opportunities is called hold-off time. The hold-off time for a node is given by the following equation:

$$HoldoffTime = 2^{TxtHoldoffExp+Base} \tag{2}$$

The base value is set to 4 according to the standard to guarantee minimum requirements for fairness. In [14], the authors set the base value to 0, and exponent value accordingly using number of two-hop neighbors and the data traffic analysis to improve the performance of mesh scheduler.

In the distributed coordinated scheduling, the control and data channels are separate and control message scheduling determines data scheduling. Each node broadcasts the range of opportunities it considers for transmissions. The nodes whose ranges of transmission opportunities overlap with their two-hop neighbors' ranges run a *pseudo-random election algorithm* in a distributed manner for each transmission opportunity. The election algorithm selects only one winner for each transmission opportunity, so that the transmissions of the control messages are collision free. Thus every node transmits at most once per control subframe. Each node runs the pseudo-random election algorithm after the hold-off time to find out the next transmission opportunity where the node is eligible to transmit a control message. The election algorithm takes the node ID, the transmission opportunity number, and the IDs of its competing nodes to find out whether the node is eligible to send a control message in that transmission opportunity or not, and returns a success if it wins, else it runs the algorithm again for next slot. The competing nodes are the nodes among its two hop neighborhood whose transmission interval coincides with this nodes transmission interval. In [2], Cao et al. propose a mathematical model for this pseudorandom election algorithm and calculate the expected number of slots between two successful transmissions as follows:

$$E[\tau] = 2^{x\,+\,\mathrm{base}} + E[S], \tag{3}$$

where $E[S]$ is the expected number of contention slots, x is its TxtHoldoffExp, and $E[\tau]$ is the number of transmission opportunities between two successful transmissions, which is equal to the summation of hold-off time and contention time. $E[S_k]$ for node k is given by the following equation:

$$E[S_k] = \sum_{j=1, j\neq k, x_j \geq x_k}^{\aleph_k^{\mathrm{known}}} \frac{2^{x_j} + E[S_k]}{2^{x_j + \mathrm{base}_j} + E\left[S_j\right]} + \left(\sum_{j=1, j\neq k, x_j < x_k}^{\aleph_k^{\mathrm{known}}} 1 \right) + \aleph_k^{\mathrm{unknown}} + 1, \quad \text{where } k = 1, \ldots, N. \tag{4}$$

Here N denotes the number of nodes in the network, $\aleph_k$ denote the set of 2-hop neighbor nodes of node k. $\aleph_k^{\mathrm{unknown}}$ is

the set of nodes whose scheduling information are unknown in the neighbor nodes set $\aleph_k$. Let $\aleph_k^{\text{known}} = \aleph_k / \aleph_k^{\text{unknown}}$. The above equation determines the expected number of slots between two successful transmissions and can be solved using a fixed point iteration. It is to be noted that the Hold-off exponent is an important parameter in mesh distributed scheduling. It determines not only the hold-off time but also the expected number of slots between two successive transmissions. If the exponent is increased, the hold off time increases whereas the expected number of competitors of a node decreases and hence probability of success increases. Decreasing the exponent value leads to the opposite situation.

4. Proposed Scheme

In this section we give a detailed description of our proposed *expected scheduler delay* (ESD) metric for delay sensitive routing, and a distributed source routing algorithm which makes use of Algorithm 2 to route the packets through minimum delay path across the network in a loop-free manner.

4.1. Proposed Routing Metric—Expected Scheduler Delay (ESD). Let $G(N, E)$ be a network graph where N is the set of network nodes, and E is the set of edges between the nodes. Let, X be an intermediate node in the network. We define the parameter *flow descriptor (flowdesc)* at node X as follows: $flowdesc_X$ = total number of incoming and outgoing MAC level flows at node X.

Here MAC level flow means a set of packets with the similar source MAC address and the similar destination MAC address.

We assume each node knows the *flowdesc* of all the nodes in its two-hop neighborhood. Later we describe how it can learn this parameter. Now consider two nodes A and B who are directly connected. Now the expected scheduler delay (ESD) from node A to B is defined as follows:

$$\text{ESD} = \frac{\left(E[\tau_A] \times flowdesc_A + E[\tau_B] \times flowdesc_B\right)}{2}. \quad (5)$$

That is the ESD of a link is the average of the expected waiting time at two ends on the link. Here we use the metric *flowdesc* to avoid congestion at a single path. The metric actually takes care of number of flows currently being served by a node. As IEEE 802.16 mesh used a fair round robin scheduler, so it is justified to multiply the *flowdesc* with expected waiting time, and then taking the average. The divide by two ensures that the delay is equally divided among the two endpoint nodes on a link.

4.2. A Distributed Mechanism to Calculate ESD. In IEEE 802.16 mesh mode scheduling the MSH-DSCH messages are used to carry the distributed scheduling information. These MSH-DSCH messages are scheduling messages and are broadcasted in the network. In the distributed scheduling mode, each node competes with its 2-hop neighbors for channel access using a pseudorandom election algorithm based on the scheduling information of its two-hop neighborhood. To inform *flowdesc* and $E[\tau]$ values to all other nodes, each node uses MSH-DSCH messages. Here each node maintains a routing table that contains current *flowdesc* and $E[\tau]$ values of all other nodes. Each node includes this table information into MSH-DSCH message. This can be seen as a modified link state routing, where we broadcast the table information instead of link state table. To reduce the size of the table, each node broadcasts only the updated information, and the timestamp of last update. Thus once all the node state informations are updated, the nodes use Algorithm 2 to find out the minimum delay path to the destination. This is a source routing mechanism, where the source finds out the shortest path to the destination, and appends this path information to the packets. Each node uses the Bellman-Ford algorithm to find out the shortest path. To compute the shortest path from Bellman-Ford algorithm, each node constitutes the network graph, with the link cost as the ESD. As the value of ESD is nonnegative, the routing path is guaranteed to be loop free.

Here we only consider the effective delay of each node. If the effective delays at nodes P and Q are $E[\tau_P]$ and $E[\tau_Q]$, then the link delay between P and Q is calculated as the average of the effective delays at P and Q. The delay of a path is taken as the sum of all the link delays along that path. We have used the Bellman-Ford algorithm for routing in the network. However, this Bellman-Ford algorithm differs from the traditional Bellman-Ford algorithm which uses hop-count as the routing metric. Here the routing metric is based on the effective delay at each node. The effective node delay of all the nodes in the network are stored in a data structure. These effective delays change over time as packets flow among the nodes in the network and are used for routing packets across the network.

Each node executes Algorithm 1 for initialization. The variables and metrics used for the algorithm is shown in Table 2. In this initialization part each node maintains 3 fields. First field is *connected* (i, j)—it defines whether node i is connected to node j or not. Second field is *distance*(i, j)—it defines the distance between node i and node j. It is set to ∞ if there is no connection between node i and node j. If node i and node j are connected then *distance*(i, j) is set to 1. If node i and node j are the same node then *distance*(i, j) is set to 0. Third field is *delay*(i, j)—it defines the ESD of link between node i and node j. If these two nodes are not connected then this field is set to ∞. If connected then, set to actual link ESD and if these two nodes are equal then set to 0.

Next we present Algorithm 2 that routes the packets across the network. This algorithm finds path of minimum ESD and routes the packets across that path. In the first outer for loop of algorithm path vector is initialized. This path vector stores the node through which packets are transferred to achieve minimum ESD. If ESD between two nodes is greater than 0 then corresponding path between those two nodes contains the starting node. Thus paths between source and destination nodes are created.

The performance analysis of the ESD metric is done in the subsequent simulation results section in comparison with the hop count metric.

Table 2: State variables at each node i.

Variable name	Type	Meaning
$distance(i, j)$	Number	Distance between nodes i and j
$delay(i, j)$	Number	Delay of the link connecting nodes i and j
$path(i, j)$	Set	Path between nodes i and j
$connected(i, j)$	Boolean	Returns *TRUE* if i, j are connected, else returns *FALSE*
ESD(i, j)	Number	ESD of the link connecting nodes i and j
$\mathbb{N}$	Number	Number of nodes in the network

```
(1) for i ← 1 to ℕ do
(2)       for j ← 1 to ℕ do
(3)          if (i ≠ j) ∧ (¬connected (i, j)) then
(4)             distance(i, j) ← ∞
(5)             delay(i, j) ← ∞
(6)          else if connected(i, j)then
(7)             distance(i, j) ← 1
(8)             delay(i, j) ← ESD(i, j)
(9)          else
(10)            distance(i, j) ← 0
(11)            delay(i, j) ← 0
(12)         end if
(13)      end for
(14) end for
```

Algorithm 1: Initialization.

4.3. Correctness of ESD. The ESD metric is designed in such a way that it takes into consideration the delay requirements of delay sensitive traffic flows. ESD reflects the time gap between two successive transmissions of a node in the coordinated, distributed mesh scheduler. This implies that the delay modeled by ESD is in terms of the transmission opportunities (TxOpp) or slots. To verify the correctness of ESD, it is required to show that the delay modeled by ESD is capable of reflecting the actual end-to-end delay of the routing paths. We have conducted simulations in order to establish this fact. We have taken 3×3, 4×4, and 5×5 grid topology for the simulations. We have used the two farthest nodes along the diagonal of grid as source and destination. For all the grid sizes a fixed number of traffic flows were considered. For each grid size we have calculated the delay as given by ESD, in terms of slots and also found out the generated end-to-end delay, in terms of milliseconds. The resultant performance graph is shown in Figure 1. In this figure we compare delay between ESD and end-to-end path delay. This graph shows that as the grid size is increased the end-to-end delay increases proportionately and so does the ESD delay. The increase in the end-to-end delay is due to the fact that as the grid size increases the distance between the source and the destination also increases, thus incurring more delay. For a similar reason the ESD delay also increases. The nature of the increase of the ESD delay is comparable to that of the end-to-end delay. Thus the figure illustrates that as the grid size increases the ESD delay increases proportionately. Hence ESD metric can reflect the actual delay in terms of slots. This implies the correctness of our metric.

```
(1) for i ← 1 to ℕ do
(2)       for j ← 1 to ℕ do
(3)          if delay(i, j) > 0 then
(4)                path(i, j) ← i
(5)          else
(6)                path(i, j) ← 0
(7)          end if
(8)       end for
(9) end for
(10) for k ← 1 to ℕ do
(11)      for i ← 1 to ℕ do
(12)         for j ← 1 to ℕ do
(13)            if delay(i, j) > delay(i, k) + delay(k, j)
                then
(14)               distance(i, j) ← distance(i, k) +
                   distance(k, j)
(15)               delay(i, j) ← delay(i, k) +
                   delay(k, j)
(16)               path(i, j) ← path(k, j)
(17)            end if
(18)         end for
(19)      end for
(20) end for
```

Algorithm 2: Computation of routing path.

5. Analysis of the Metric Expected Scheduler Delay (ESD)

To ensure good performance, a routing metric should ensure the following four properties:

(i) route scalability,

(ii) good performance for minimum weight paths,

(iii) efficient algorithm to calculate minimum weight paths,

(iv) loop-free routing.

Here we analyze each of these properties with respect to our routing metric, expected scheduler delay (ESD).

5.1. Route Scalability. To ensure route scalability, the path weights should be stable after a finite amount of time. Our metric ESD uses the mathematical estimation $E[\tau]$ and number of flows (*flowdesc*) through each neighboring nodes. The $E[\tau]$ value is a fixed value given a network topology with each node with a fixed holdoff exponent. Here *flowdesc* is

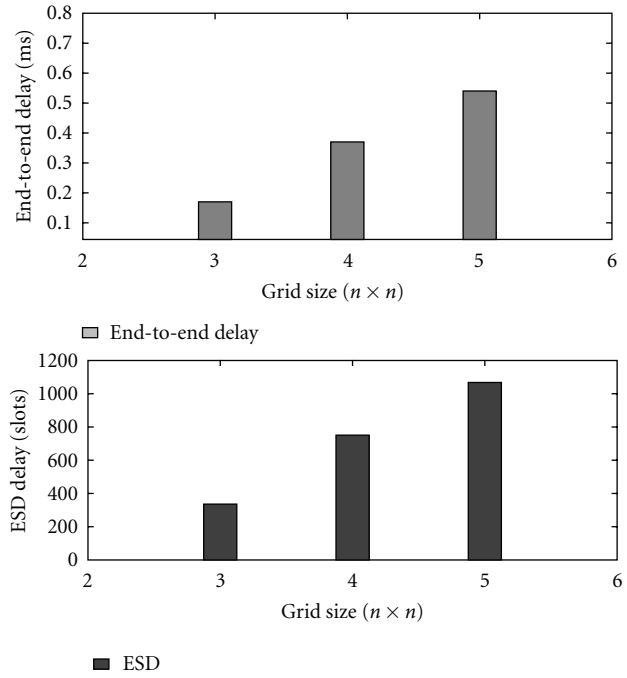

Figure 1: Expected scheduler delay versus end-to-end delay.

a variable parameter that depends on current flow path. Now in a stable network, number of flows becomes stable after a finite amount of time. So, for a fixed network topology, and fixed number of flows, the ESD value does not change over time. The ESD value changes when a new flow establishes in the network, resulting reassignment of flows based on their weights, whether they are delay sensitive or delay insensitive. So this metric is more stable than other load-sensitive metrics such as degree of nodal activity or interference switching cost.

5.2. Good Performance for Minimum Weight Paths. The ESD metric considers two important parameters while calculating minimum weight paths:

(1) scheduling delay at each node,

(2) congestion at each node.

Thus this metric tries to minimize both scheduling delay as well as congestion. In Section 4.3, it has been shown by simulation that scheduling delay directly reflects total delay. Hence by minimizing scheduling delay at each hop, total delay is minimized. Furthermore, congestion avoidance also indirectly reduces interflow interference as well as intraflow interference. So, it is expected that ESD metric will perform better which is also shown later by simulation.

5.3. Efficient Algorithm to Calculate Minimum Weight Paths. In Section 4.2, an efficient distributed algorithm based on Bellman Ford algorithm is given to compute minimum weight path between each source destination pair. A necessary and sufficient condition for the correctness of the given algorithm is that the routing metric must have a property

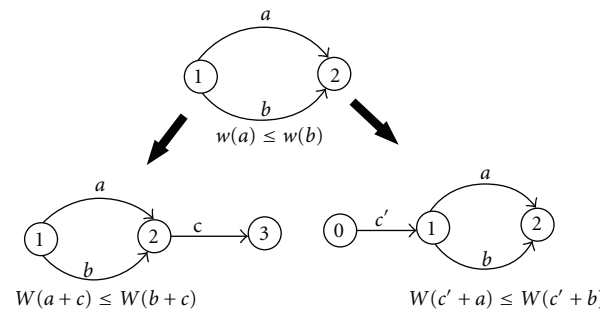

Figure 2: Example of isotonicity.

called isotonicity [17, 18]. The property can be defined as follows.

Definition 1. A routing metric $W(\cdot)$ is isotonic if $W(a) \le W(b)$ implies both $W(a \oplus c) \le W(b \oplus c)$ and $W(c' \oplus a) \le W(c' \oplus b)$, for all a, b, c, c'.

An example is shown in Figure 2. In this figure W is a routing metric. We apply it on paths a and b, between node 1 and node 2. Now $W(a) \le W(b)$. Now let c and c' be two paths leading or trailing of node 1 and node 2. To satisfy isotonicity, the following two constraints must be satisfied:

(1) $W(a + c) \le W(b + c)$,

(2) $W(c' + a) \le W(c' + b)$.

We have the following lemmas and theorems.

Lemma 2. *Let $a(i, j)$, $b(k, l)$, and $c(m, n)$ be three links where $i, j, k, l, m,$ and n are the endpoint nodes. Then the following property is satisfied.*

If

$$\mathrm{ESD}_a \le \mathrm{ESD}_b \iff (\mathrm{ESD}_a + \mathrm{ESD}_c) \le (\mathrm{ESD}_b + \mathrm{ESD}_c). \quad (6)$$

Proof. From the definition of ESD,

$$\mathrm{ESD}_a = \frac{\left(E[\tau_i] \times flowdesc_i + E\left[\tau_j\right] \times flowdesc_j\right)}{2},$$

$$\mathrm{ESD}_b = \frac{\left(E[\tau_k] \times flowdesc_k + E[\tau_l] \times flowdes_l\right)}{2},$$

$$\mathrm{ESD}_c = \frac{\left(E[\tau_m] \times flowdesc_m + E[\tau_n] \times flowdesc_n\right)}{2}. \quad (7)$$

Now,

$$\mathrm{ESD}_a \le \mathrm{ESD}_b, \quad (8)$$

so

$$\left(E[\tau_i] \times flowdesc_i + E\left[\tau_j\right] \times flowdesc_j\right) \le \left(E[\tau_k] \times flowdesc_k + E[\tau_l] \times flowdesc_l\right). \quad (9)$$

Adding $(E[\tau_m] \times flowdesc_m + E[\tau_n] \times flowdesc_n)$ to both sides of the inequality it can be easily shown that

$$(\mathrm{ESD}_a + \mathrm{ESD}_c) \leq (\mathrm{ESD}_b + \mathrm{ESD}_c). \tag{10}$$

□

Theorem 3. *The ESD metric satisfies isotonicity property.*

Proof. Consider the following:

$$\begin{aligned} & W(a) \leq W(b) \\ & \Longleftrightarrow (\mathrm{ESD}_1 + \cdots + \mathrm{ESD}_n) \\ & \leq (\mathrm{ESD}_2 + \cdots + \mathrm{ESD}_m), \end{aligned} \tag{11}$$

where there exists two paths $(1 \cdots n)$ and $(2 \cdots m)$ between nodes a and b.

So,

$$\begin{aligned} W(a \oplus c) \leq W(b \oplus c) &\Longleftrightarrow (\mathrm{ESD}_1 + \cdots + \mathrm{ESD}_n + \mathrm{ESD}_c) \\ &\leq (\mathrm{ESD}_2 + \cdots + \mathrm{ESD}_m + \mathrm{ESD}_c), \\ W(c' \oplus a) \leq W(c' \oplus b) &\Longleftrightarrow (\mathrm{ESD}_{c'} + \mathrm{ESD}_1 + \cdots + \mathrm{ESD}_n) \\ &\leq (\mathrm{ESD}_{c'} + \mathrm{ESD}_2 + \cdots + \mathrm{ESD}_m). \end{aligned} \tag{12}$$

The above two conjecture, can be easily proved by generalizing Lemma 2. □

So, from Theorem 3, it can be said that the Bellman-Ford algorithm as described in Section 4.2 can correctly compute the path between a pair of source and destination.

5.4. Loop-Free Routing. As shown in Sobrinho's work [17], a routing metric should be isotonic to ensure loop-free routing in Dijkstra or Bellman-Ford algorithm. We have the following theorem.

Theorem 4. *If Bellman-Ford's algorithm is used in hop-by-hop routing, isotonicity is a sufficient and necessary condition for loop-free routing.*

Proof. The proof directly comes from the proof given in [17] for Dijkstra's algorithm. □

As ESD metric has an isotonicity property, as proved in Theorem 3, it always produces loop-free routing.

6. Service Differentiation Using ESD

In this case we consider two types of services—delay sensitive and best effort. For service differentiation we use weighted mechanism. In the transport layer we use a bit named *serviceType* to assign a weight for each packet. Weight 1 is used for delay sensitive service and 0 is used for best effort service. When a packet is forwarded from a node to the next hop then for delay sensitive service the shortest path is calculated based on the ESD metric and for best effort service packets follow the shortest path calculated based on the hop count metric. The algorithm for service differentiation model is given in Algorithm 3. This algorithm is executed for each packet flow. For each packet the service type is already set from application file. Here this type is checked first. If the service type is 1 then this packet should be transferred using minimum delay path. To find minimum delay path Algorithm 2 must be executed. If service type is other than 1 then packet is transferred using traditional Bellman-Ford Algorithm for hop count metric.

```
(1) for all packet_j do
(2)     if serviceType = 1 then
(3)         execute Algorithm 2
(4)     else
(5)         execute Bellman-Ford Algorithm using hop-
            count
(6)     end if
(7) end for
```

Algorithm 3: Service differentiation at source.

6.1. Delay Sensitive Routing Path Computation. In the transport layer we use a bit *serviceType* = 1 for delay sensitive packet. When a packet is forwarded from the source to its next hop then check this bit value. If it is 1 then packet follows delay sensitive service path which is computed based on ESD metric. The pseudocode for ESD metric computation is already given in Algorithm 2.

6.2. Best Effort Routing Path Computation. In the transport layer we use a bit *servicetype* = 0 for best effort service packet. When a packet is forwarded from the source to its next hop then check this bit value. If it is 0 then packet follows best effort service path which is computed based on hop count metric. Similar algorithm is followed by the next hop to calculate its next hop. The shortest path based on the Hop Count metric is calculated using the traditional Bellman-Ford algorithm.

7. Simulation Results

The simulations are done using the NS2 [19] simulator. We use the ns2mesh80216-2.33-081113 patch [20] for NS-2.33 to simulate IEEE 802.16 mesh network. We have taken a 4×4 grid topology. We choose grid topology as we know that mesh is a type of networking where each node must not only capture its own data, but also relay those data for other nodes, that is, in the network all nodes should be connected. If we use square grid topology then the number of possible routes from source mesh node to destination node easily increases and all of these routes share a combination of links thus probability of connectivity among nodes increases. Multipath routing is important for a mesh node to have a second, nonoverlapping route to the destination. Square grid topology gives better coverage than random topology, and

Table 3: Simulation Environment.

Parameter	Value
Number of channels	1
Number of radios	1
Channel PDU error rate	0
Channel bandwidth	10 MHz
Hold-off exponent	0
Bandwidth manager	Fair round robin
Packet scheduler	Fair round robin
Buffer size	1000000 bytes
CBR packet size	1000 bytes
CBR traffic rate	2 Mbps
TCP packet size	1024 bytes
TCP CW_{Max}	64

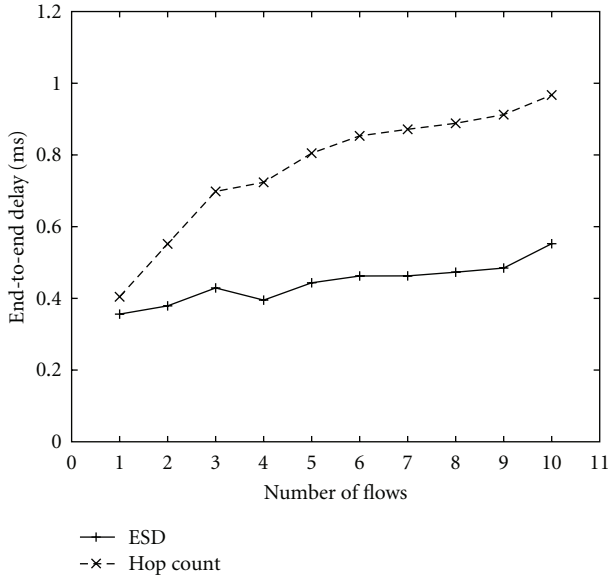

Figure 3: Delay comparison: ESD metric versus hop count.

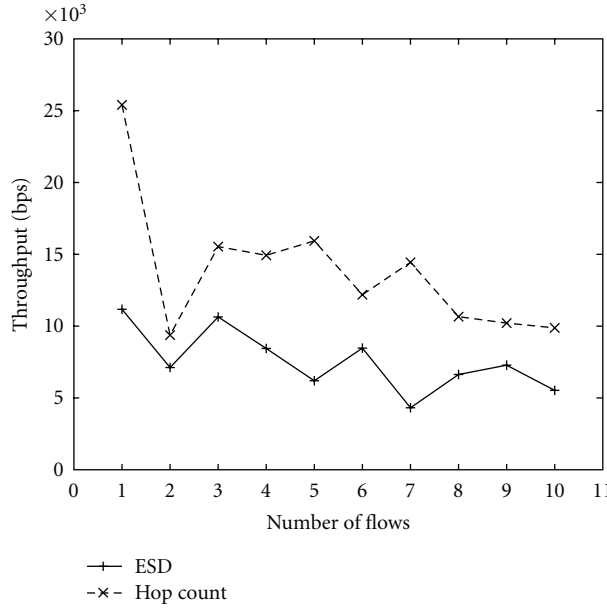

Figure 4: Throughput comparison: ESD metric versus hop count.

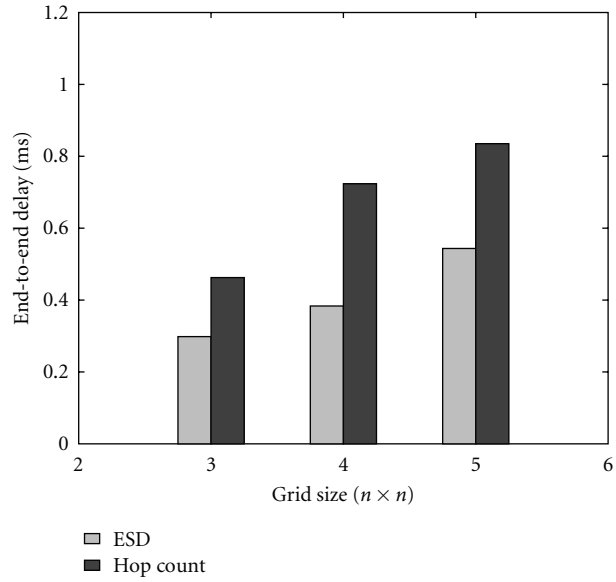

Figure 5: Delay comparison: ESD metric versus hop count.

the output can easily be analyzed. The additional loss in performance of a random network is due to the double wireless resource demands in gateways. The reason for this demand is that there is a probability that two wired gateways interfere with each other, but it does not increase the number of clients served. In square grid topology network is more organized, so this problem does not arise. An overview of simulation environment is given in Table 3. The hold-off exponent value is chosen as 0, because, for static hold-off exponents, the delay is minimized with hold-off exponent 0, as shown in [14, 15]. Furthermore, the traffic type is taken as FTP over TCP traffic for all the simulations except where it is explicitly mentioned.

Two nodes at two opposite corners of a diagonal of the grid are taken as source and destination. While all the packets of all the flows follow the same 3-hop path from source to destination, and thus increase the end-to-end delay just for network congestion in case of standard routing using hop-count, our proposed scheme chooses the most suitable path having the minimum delay. Figures 3 and 4 give the delay (in ms) and throughput comparison between our proposed ESD metric and hop count metric. These figures show that our scheme performs better than the hop-count metric specially with high loads. From these graphs it is clear that our proposed scheme can effectively route delay sensitive service packets along the path having minimum delay. To show the effect of grid topology on our ESD metric we consider 3×3, 4×4, and 5×5 grid topology. Then for 10 flows we compare the delay for ESD and hop count metrics and the result is shown in Figure 5. From this figure it is clear that our ESD metric performs better than traditional hop count metric for higher load of traffic.

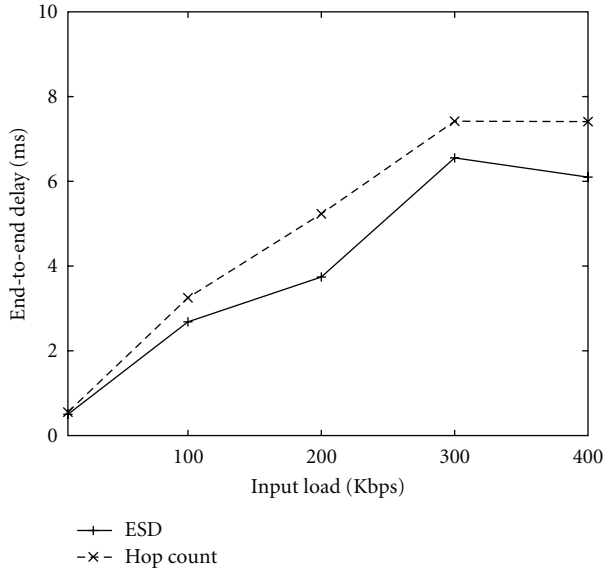

Figure 6: Scalability in terms of input load (for TCP traffic).

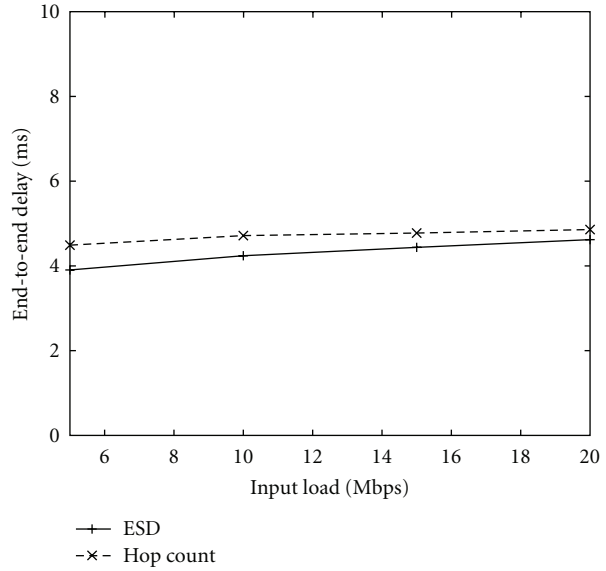

Figure 7: Scalability in terms of input load (for UDP traffic).

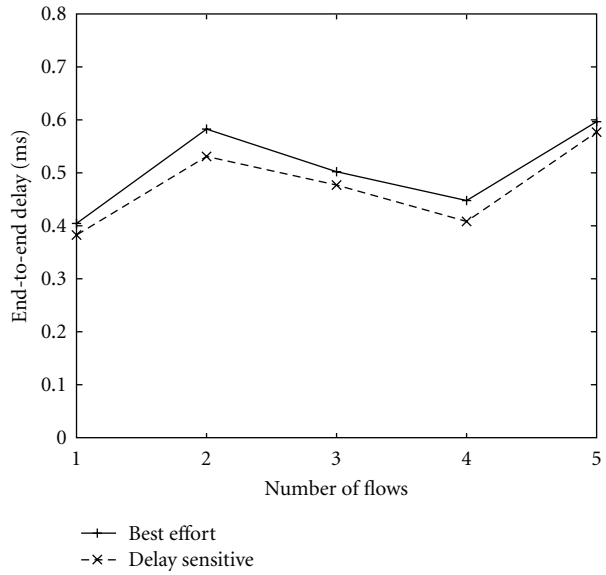

Figure 8: Delay comparison: delay sensitive versus best effort services.

To show the scalability of the proposed ESD metric, Figure 6 shows the end-to-end delay for FTP over TCP traffic with respect to input load. As the input load increases, the proposed ESD metric performs better than standard hop-count-based routing algorithm. Figure 7 shows the same for CBR over UDP traffic. For UDP traffic also, the proposed ESD metric performs better than hop-count-based routing.

Now for differentiated service model we use a 4×4 grid topology and consider a total of 10 flows—5 flows for delay sensitive traffic and 5 flows for best effort traffic. Here also we consider two nodes at the two farthest corners of a diagonal as the source and destination nodes. Figure 8 depict the end-to-end delay comparison between delay sensitive and best effort services with holdoff exponent 0. For a particular constant value of HoldOffExp as the number of flows is increased more packets travel across the network, consequently increasing the *flowdesc* of the nodes in the network. Hence the delay also increases proportionately with the number of flows. It is seen that as the number of flows increases the delay for both cases increases. However, the delay experienced by delay sensitive service packets which routes through the shortest path calculated based on ESD metric is less than that experienced by the best effort service packets which routes through the shortest path calculated based on hop count metric. Thus ESD chooses the path having minimum delay for delay sensitive flows.

In Figure 9 we have shown a comparison of transport layer end-to-end throughput between delay sensitive and best effort traffic with hold-off exponent 0.

To show the effect of grid topology in our differentiated service model we use 2×2, 3×3, 4×4, 5×5, and 6×6 grids. We take one flow for each type of service and compare the delay and throughput for delay sensitive and best effort flows. In Figure 10 we have shown a comparison of end-to-end delay between delay sensitive and best effort traffic with hold-off exponent 0. This graph shows that with increasing grid size the delay for both the service types increases. In most of the cases the delay for the delay sensitive flow is lower than that of the best effort flow. However as best effort Service chooses the path with highest bandwidth so it may sometimes happen that this path delay is less than delay sensitive Service path delay. That is why in this figure for grid 2×2, 3×3, and 4×4 best effort service gives a better result than delay sensitive service. This implies that in the differentiated service architecture, the ESD metric successfully routes packets having delay constraints, that is, packets which should travel across the network experiencing as low delay as possible along paths with minimum delay. Figure 11 shows

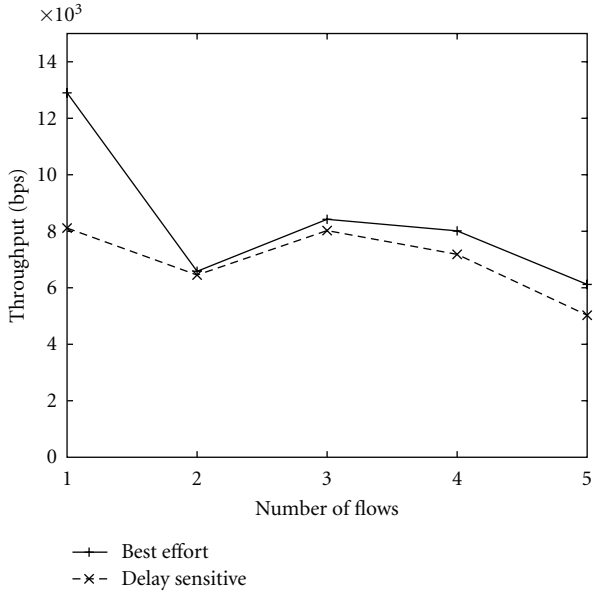

Figure 9: Throughput comparison: delay sensitive versus best effort services.

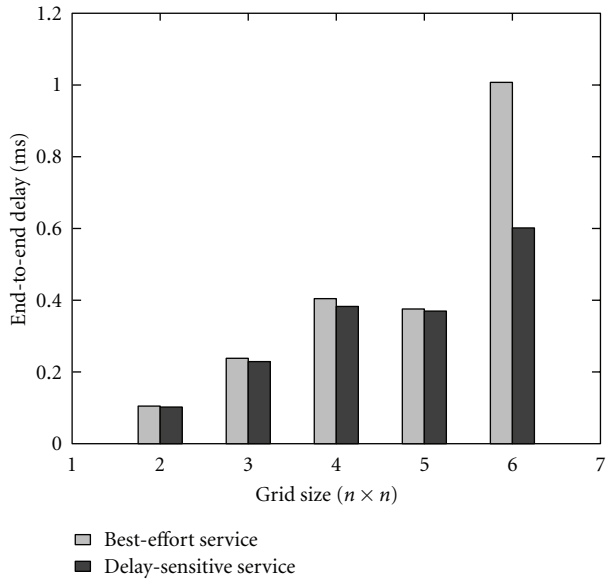

Figure 10: Delay comparison: delay sensitive versus best effort.

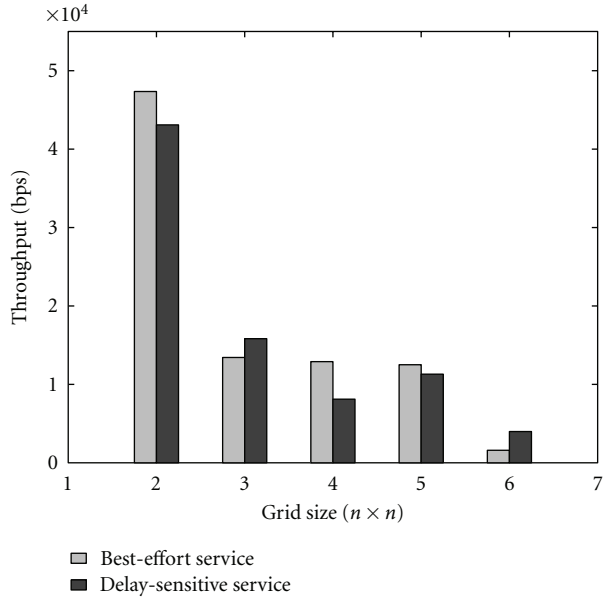

Figure 11: Throughput comparison: delay sensitive versus best effort.

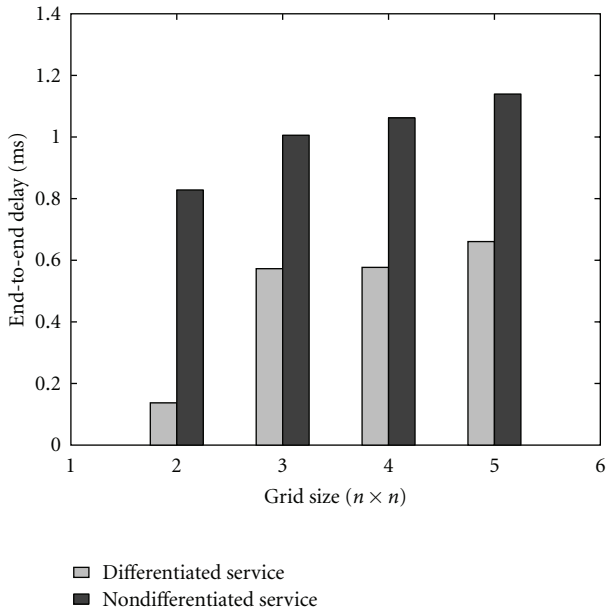

Figure 12: Delay comparison: differentiated and non-differentiated service for delay sensitive flow.

the throughput comparison of these two service types for different grid sizes.

For service differentiation overall delay to route a packet is lower than delay in non differentiated architecture, specially for the delay sensitive services. To show this effect we compare the delay for delay sensitive Service and best effort service of DiffServ model with non differentiated service model for 2×2, 3×3, 4×4, and 5×5 grid topology. The DiffServ architecture gives better result for each topology. The result is shown in Figure 12. In case of best effort service it gives better result than non differentiated service architecture because of lower number of traffic flows, hence the congestion is low. The behavior is shown in Figure 14. As in the previous cases, the cost here is the throughput, as depicted in Figure 13 for delay sensitive services and Figure 15 for best effort services.

8. Conclusion

This paper gives a new cross layer routing metric which provides an effective route for delay sensitive services in IEEE 802.16 mesh distributed networks. A detailed analysis of this

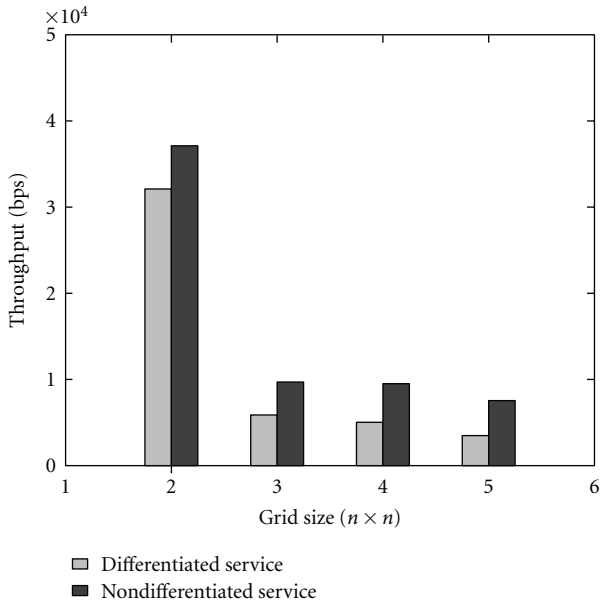

Figure 13: Throughput comparison: differentiated and non-differentiated service for delay sensitive flow.

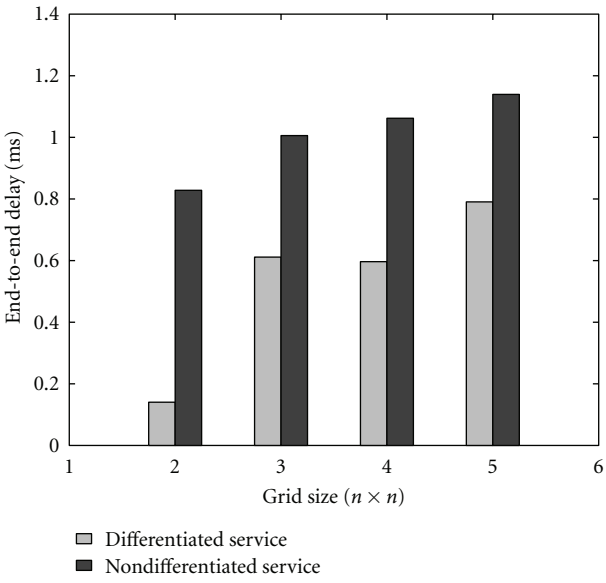

Figure 14: Delay comparison: differentiated and non-differentiated service for best effort service.

routing metric has been carried out using NS2 simulations. The implementation of this metric vindicates the theoretical analysis. We find that our new *expected scheduler delay (ESD)* metric is better compared to hop count metric in terms of delay for the service model that contains both delay sensitive and best effort service. Our proposed metric can differentiate the routing path according to the requirement for mixed service model, that is, a service model containing both delay sensitive and best effort service. So, it is expected that this *expected scheduler delay (ESD)* metric will be useful to design

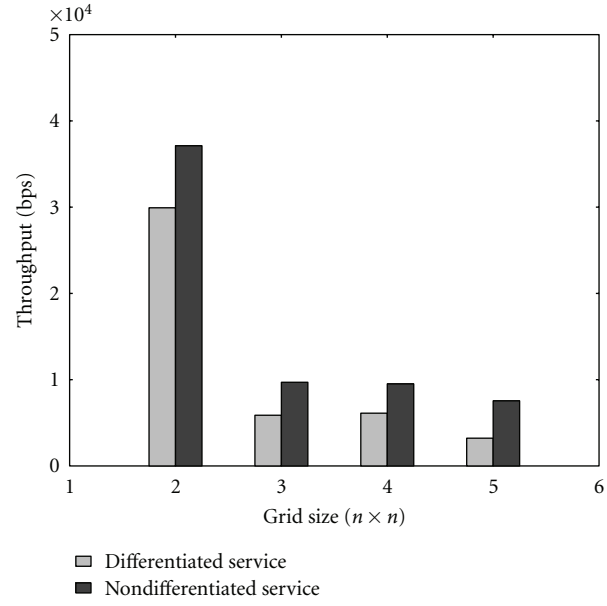

Figure 15: Throughput comparison: differentiated and non-differentiated service for best effort service.

QoS-aware routing protocols for IEEE 802.16 mesh networks.

References

[1] IEEE 802.16e/D5-2004, "Part 16: air interface for fixed and mobile broadband wireless access systems—amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands," 2004.

[2] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modeling and performance analysis of the distributed scheduler in IEEE 802.16 Mesh Mode," in *Proceedings of the 6th ACM MobiHoc*, Urbana-Champaign, Ill, USA, 2005.

[3] K. Wongthavarawat and A. Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems," *International Journal of Communication Systems*, vol. 16, no. 1, pp. 81–96, 2003.

[4] C. Cicconetti, I. F. Akyildiz, and L. Lenzini, "FEBA: a bandwidth allocation algorithm for service differentiation in IEEE 802.16 mesh networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 884–897, 2009.

[5] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks," in *Proceedings of the 1st ACM International Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP '05)*, pp. 140–149, Montreal, Canada, 2005.

[6] Y. Li, Y. Yang, and C. Cao, "A novel routing algorithm in distributed IEEE 802.16 mesh networks," *IEEE Communications Letters*, vol. 13, no. 10, pp. 761–763, 2009.

[7] T.-C. Tsai and C.-Y. Wang, "Routing and admission control in IEEE 802.16 distributed mesh networks," in *Proceedings of the 4th IEEE and IFIP International Conference on Wireless and Optical Communications Networks, (WOCN '07)*, Singapore, 2007.

[8] X. Xiao and L. M. Ni, "Internet QoS: a big picture," *IEEE Network*, vol. 13, no. 2, pp. 8–18, 1999.

[9] H. Jiang, W. Zhuang, X. Shen, A. Abdrabou, and P. Wang, "Differentiated services for wireless mesh backbone," *IEEE Communications Magazine*, vol. 44, no. 7, pp. 113–119, 2006.

[10] T. Tung, Z. Jia, and J. Walrand, "A practical approach to QoS routing for wireless networks," in *Proceeding of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '05)*, vol. 2005, pp. 286–293, Riva del Garda, Italy, 2005.

[11] Y. Wang, J. Wung, and R. Kravets, "Designing routing metrics for Mesh networks," in *Proceedings of the IEEE Workshop on Wireless Mesh Networks, (WiMesh '05)*, 2005.

[12] M. E. M. Campista, P. M. Esposito, I. M. Moraes et al., "Routing metrics and protocols for wireless mesh networks," *IEEE Network*, vol. 22, no. 1, pp. 6–12, 2008.

[13] N. Bayer, B. Xu, V. Rakocevic, and J. Habermann, "Improving the performance of the distributed scheduler in IEEE 802.16 mesh networks," in *Proceedings of the IEEE 65th Vehicular Technology Conference (VTC '07)*, pp. 1193–1197, April 2007.

[14] S. Y. Wang, C. C. Lin, H. W. Chu, T. W. Hsu, and K. H. Fang, "Improving the performances of distributed coordinated scheduling in IEEE 802.16 mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2531–2547, 2008.

[15] S. Chakraborty, D. K. Sanyal, A. Chakraborty, A. Ghosh, S. Chattopadhyay, and M. Chattopadhyay, "Tuning hold off exponents for performance optimization in IEEE 802.16 Mesh Distributed Coordinated Scheduler," in *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE '10)*, Singapore, 2010.

[16] I. Bhakta, S. Chakrabory, B. Mitra, D. K. Sanyal, S. Chattopadhyay, and M. Chattopadhyay, "Designing an efficient delay sensitive routing metric for IEEE 802.16 Mesh networks," in *Proceedings of the International Conference on Wireless and Optical Communications*, 2011.

[17] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM '01)*, pp. 727–735, April 2001.

[18] J. L. Sobrinho, "Network routing with path vector protocols: theory and applications," in *Proceedings of the ACM SIGCOMM*, pp. 49–60, August 2003.

[19] "NS2 network simulator," http://isi.edu/nsnam/ns/.

[20] The mesh extension for NS2, "Ns2mesh80216," http://cng1.iet.unipi.it/wiki/index.php/Ns2mesh80216.